Microcoils for magnetic field mapping at submillimeter, subppm resolutions

Yiqiao Ray Tang, Shin Utsuzawa, Yi-Qiao Song
Schlumberger-Doll Research, MA 20139

We built a proton procession magnetometer (PPM)[1] using a microcoil in conjunction with a motorized three-axial translation stage for mapping magnetic field distribution inside two Halbach magnets, at submillimeter and subppm resolutions. For each spatial scanning pattern, maps of the temperature-corrected field strength and local field homogeneity were generated. In this report we document the hardware setup and algorithm for this mapping technique.

The microcoil was made of 40 AWG wire, tightly wound over 1 mm in length around a glass capillary of 1 mm OD and 0.8 mm ID. In total 0.5 uL fluid volume was under investigation during an NMR experiment, as shown in Figure 1(A). Two plastic tubes were connected on both ends of the glass capillary, through which water sample was injected. The flowline was placed on a specially-made PCB board, positioned on an optical mount attached to the translation stage. The translation stage was driven by three Newport CONEX-TRA12CC actuators,[2] with spatial resolution 1.5 μm in each dimension.

To map the field distribution inside a magnet, we need to coordinate placement of the microcoil with NMR data acquisition, as shown in Figure 1(B). A Matlab program[3] was developed to control both the motorized actuators through serial ports and the KEA spectrometer through a backdoor program. During data acquisition, the stage scanned through a predefined table. After each stage movement, two NMR scans were executed and the data were saved. At each position, the Fourier-transformed spectrum is fitted by a Lorentzian function, $1/(4(f-f_0)^2+\Gamma^2)$, where f_0 is Larmor frequency and Γ is the peak width at half height. The ratio Γ/f_0 characterizes the local field homogeneity.

To map the spatial field strength, we need to characterize temperature drift of the system during the course of data acquisition. In the predefined scan pattern, the stage was translated back to the original position periodically, at which the acquired data were used for tracking temperature change.

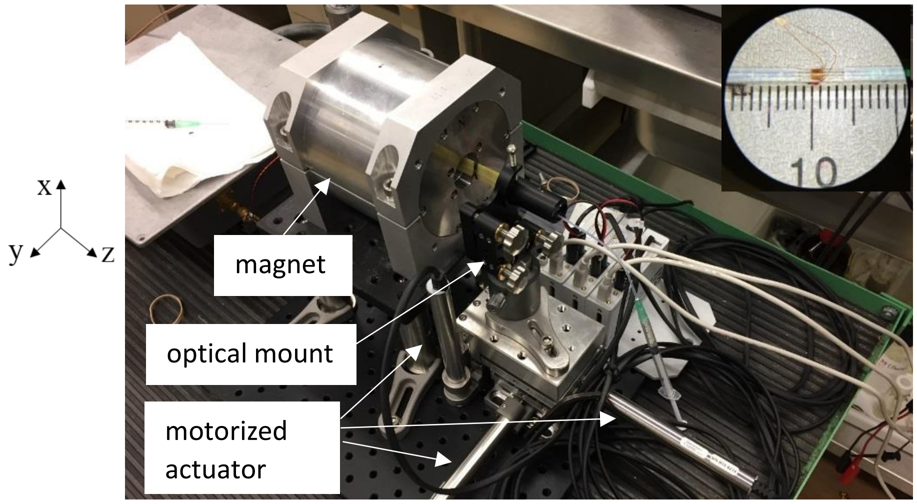
Assuming a small correction on the field strength due to temperature drift, $\Delta(t)$, where t is time, with a boundary condition $\Delta(0) = 0$. From the periodic measurement at the initial position and linear interpolation, $\Delta(t)$ may be obtained through the entire measurement. Subsequently, $f(t) = f_0\times(1+\Delta(t))$ is determined, where $f(t)$ is the temperature-corrected field strength and f_0 is the fitting result at coordinate $(x(t), y(t), z(t))$.

Figure 2 - 5 show the mapping results for the two magnets. In both measurements, a cell of $1.2\times1.2\times1.2$ mm^3 was scanned at a step size of 0.3 mm along each dimension. The scan started at center of the cuboid, and returned to the initial position every 6 points. In total, 151 points were scanned, including 125 at evenly-spaced coordinates, and 26 at the center for temperature correction. Figure 2(4) shows the Lorentzian fitting of all FFT spectrums and the map of Γ/f_0 ratio for magnet M2 (M3). Figure 3(5) shows the temperature-correcting procedure and the field map for M2(M3). Within the scanned region, the best homogeneity is determined at 2.6 ppm for M3 and 9.9 ppm for M2. Assuming a temperature coefficient of 400 ppm/°C, a typical number for SmCo magnet, the maximum temperature drift during the one-hour measurement is on the order of 0.05 °C.

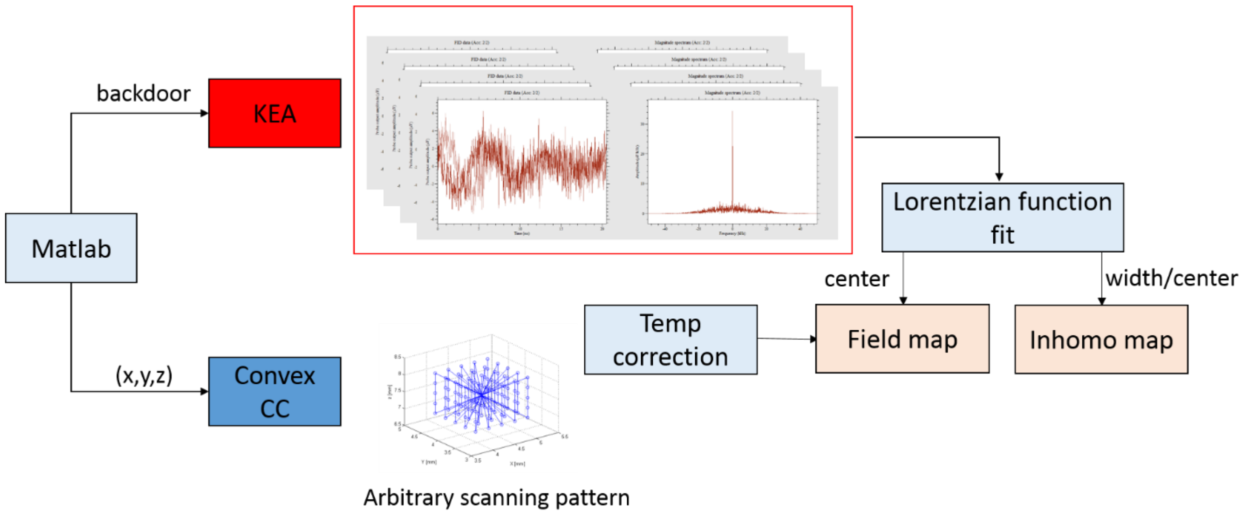
We also scanned a large cell within M2, and show the results in Figure 6 - 7. The scan pattern is a set of equilateral triangles separated by 0.6 mm along z direction, spanning over 8 mm. The local field homogeneity at each scanned point varies from 9.5 to 38 ppm. The temperature-corrected field strength increases from 23.025 MHz to 23.027 MHz, with the total fractional change 72 ppm.

We note that resolution of the PPM is dictated by a multitude of factors, such as physical size of the microcoil probe, temperature-correction quality, probe bandwidth, SNR, T_2/T_1 of the testing fluid, dwell time of acquisition, and field strength and variance within the scanned region. The magnetometer can determine proton resonance frequency within a range of 20 Hz, translating to a subppm resolution for a 0.5 T magnet. We conclude that the PPM suffices for charactering spectroscopy-graded permanent magnets.

(A)



(B)



(C)

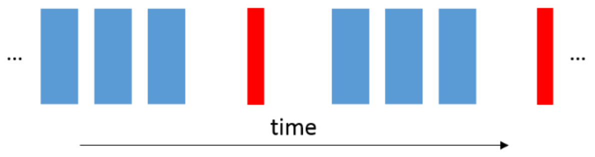
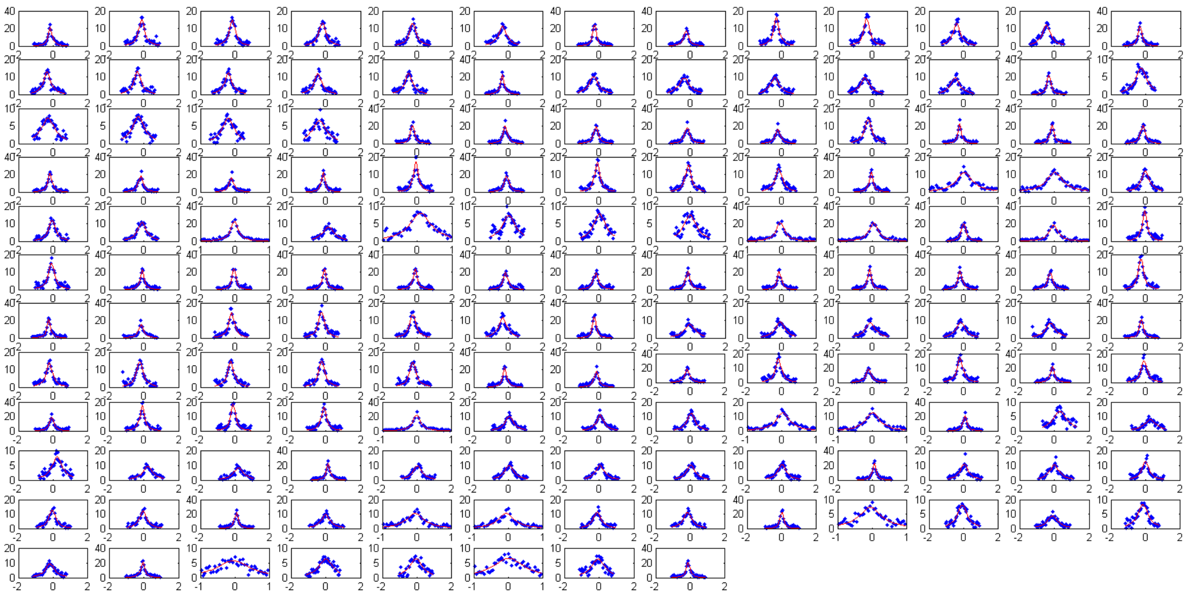


Figure 1(A) A photograph of the hardware setup and the microcoil (inset); (B) Data acquisition and analysis workflow; (C) A schematic of time sequence for motorized actuators (blue) and NMR data acquisition (red).

(A)



(B)

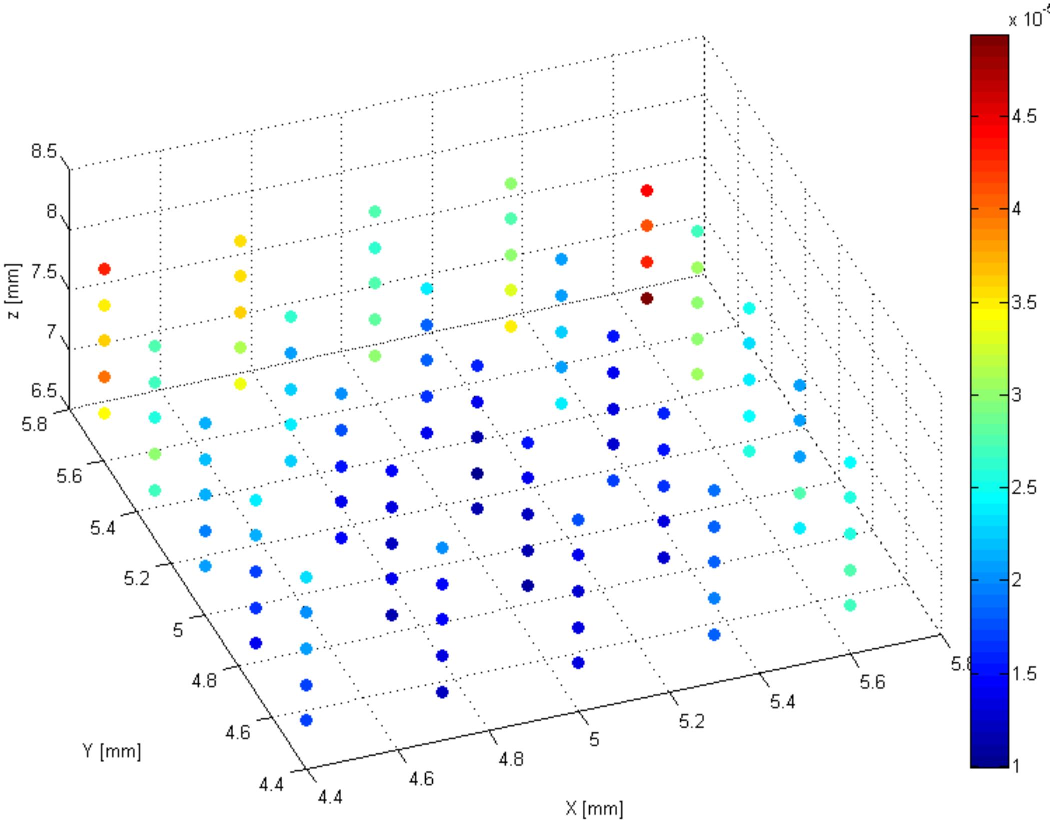


Figure 2. Field mapping for magnet M2. (A) fitting of Lorentzian function (red) to the FFT spectrum (blue) at each scanned point; (B) the map of ratio Γ / f_0.

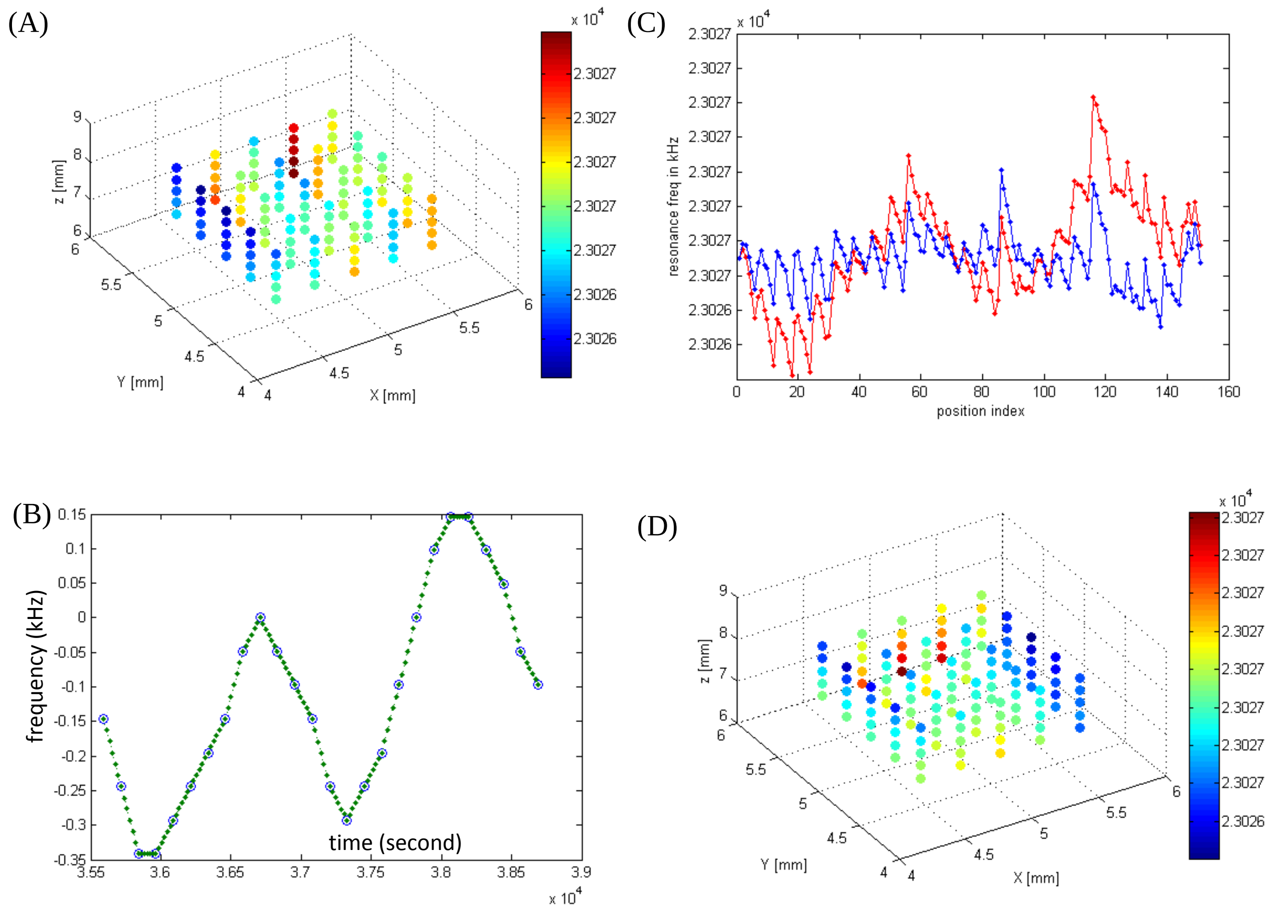
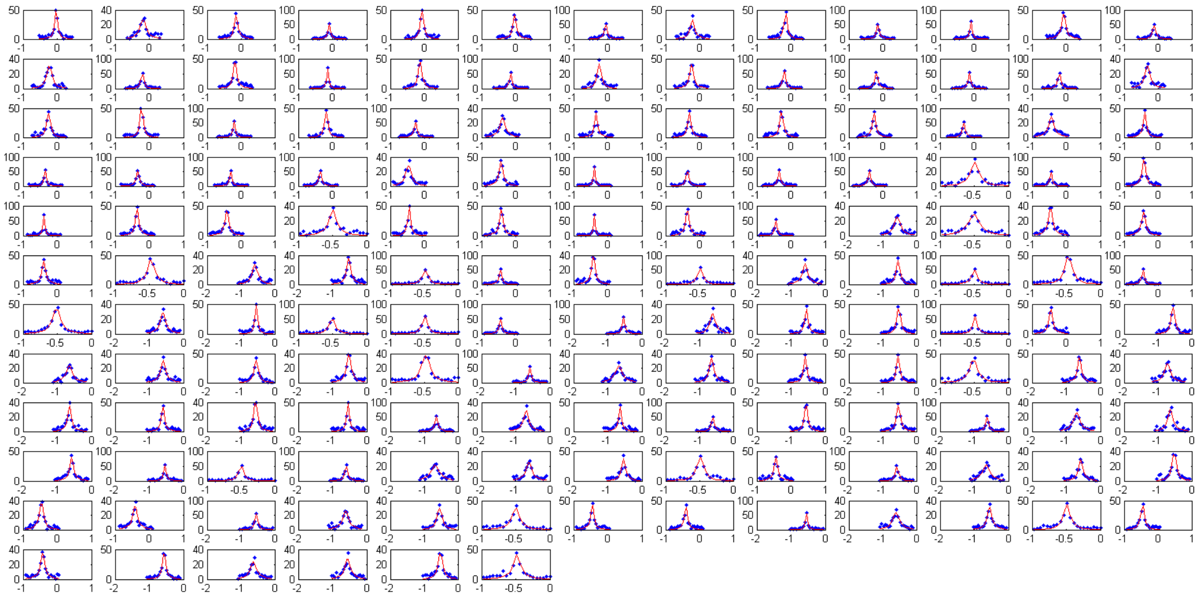


Figure 3. Field mapping for magnet M2. (A) a map of f_0 from the Lorentzian-function fitting at all scanned points; (B) Measured (blue) and linearly interpolated (green) Larmor frequency at the reference point as a function of acquisition time; (C) Raw Larmor frequency (red) and temperature-corrected frequency (blue); (D) A map of temperature-corrected f_0 from the Lorentzian-function fitting at all scanned points. Field strength presents a saddle point distribution (in xy plane) within the scanned region.

(A)



(B)

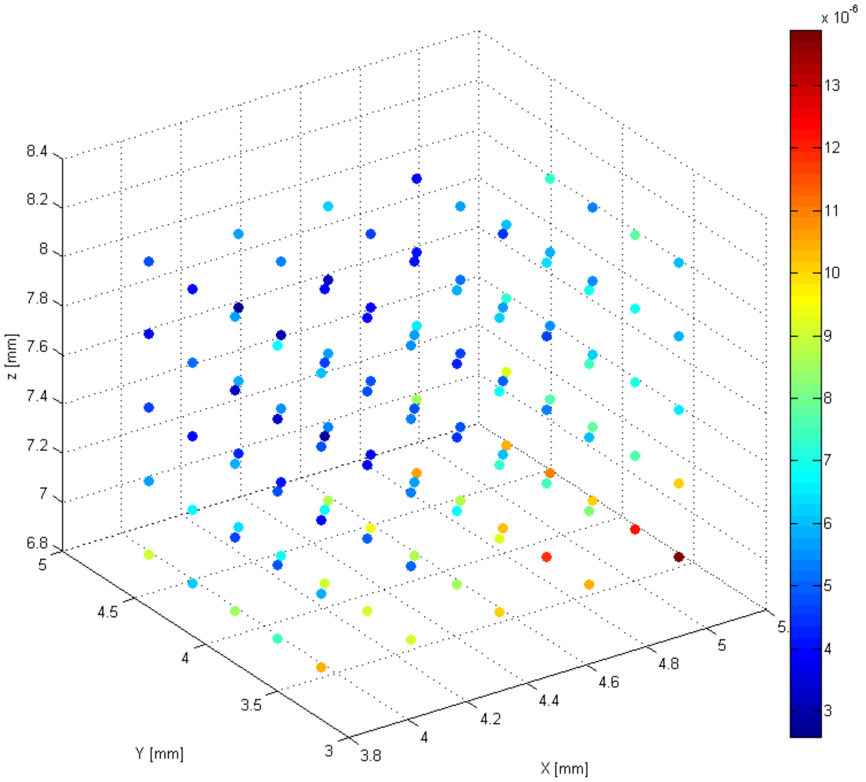


Figure 4. Field mapping for magnet M3. (A) fitting of Lorentzian function (red) to the FFT spectrum (blue) at each scanned point; (B) the map of ratio Γ/ f_0.

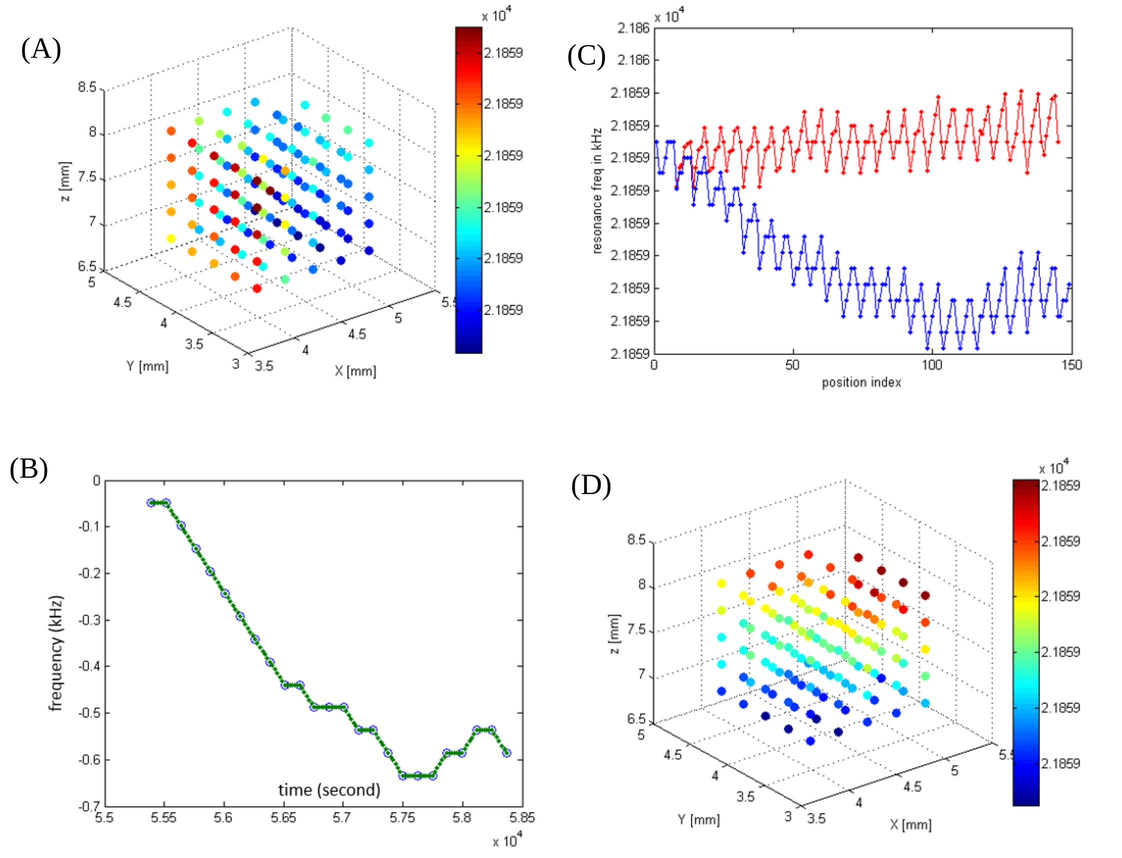


Figure 5. Field mapping for magnet M3. (A) a map of f_0 from the Lorentzian-function fitting at all scanned points; (B) Measured (blue) and linearly interpolated (green) Larmor frequency at the reference point as a function of acquisition time; (C) Raw Larmor frequency (blue) and temperature-corrected frequency (red); (D) A map of temperature-corrected f_0 from the Lorentzian-function fitting at all scanned points. The field strength varies monotonically along all three dimensions in the scanned region.

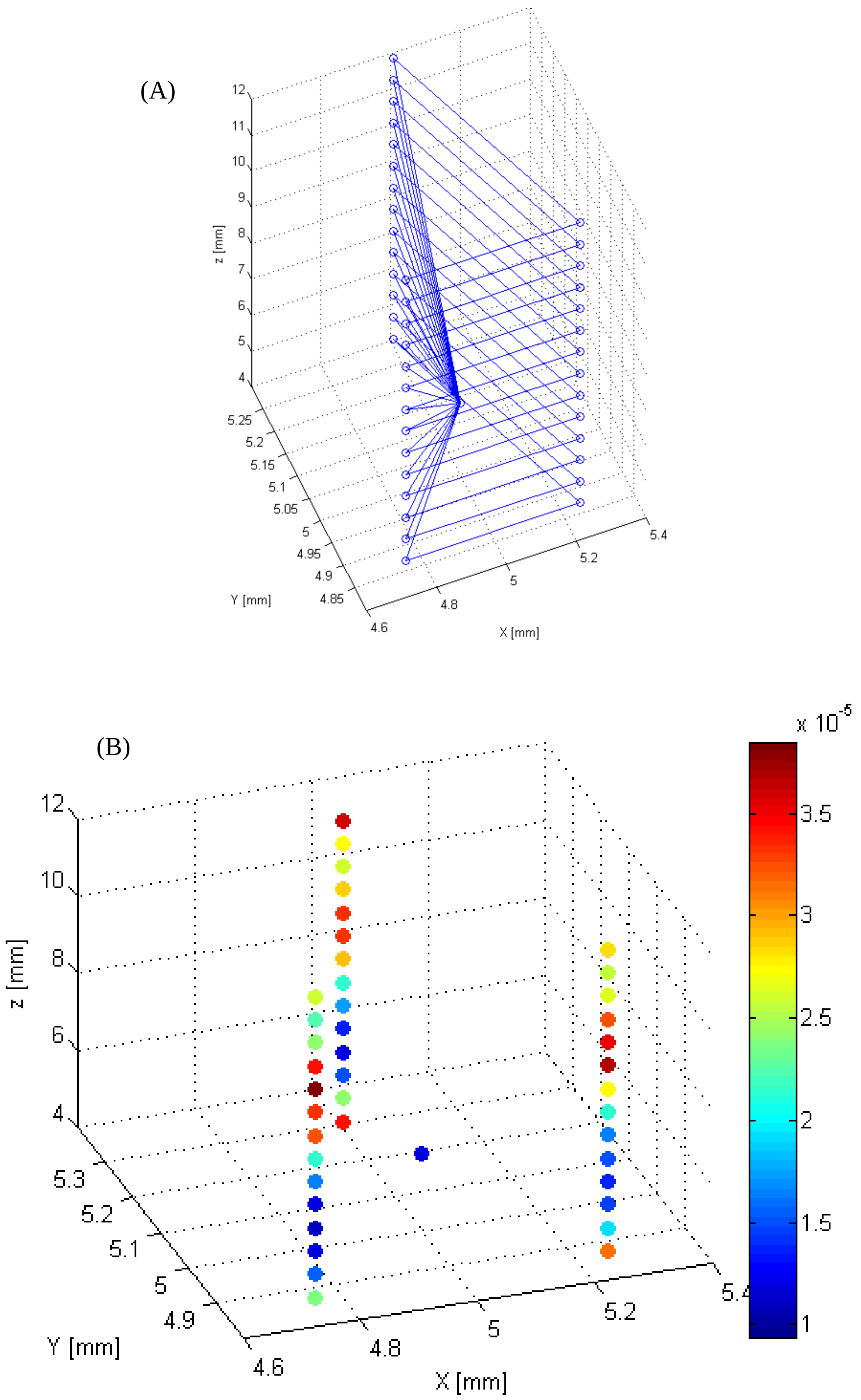


Figure 6(A) Scan patterns consists of a set of equilateral triangles separated by 0.6 mm along z direction, spanning over 80 mm; (B) the map of ratio Γ / f_0.

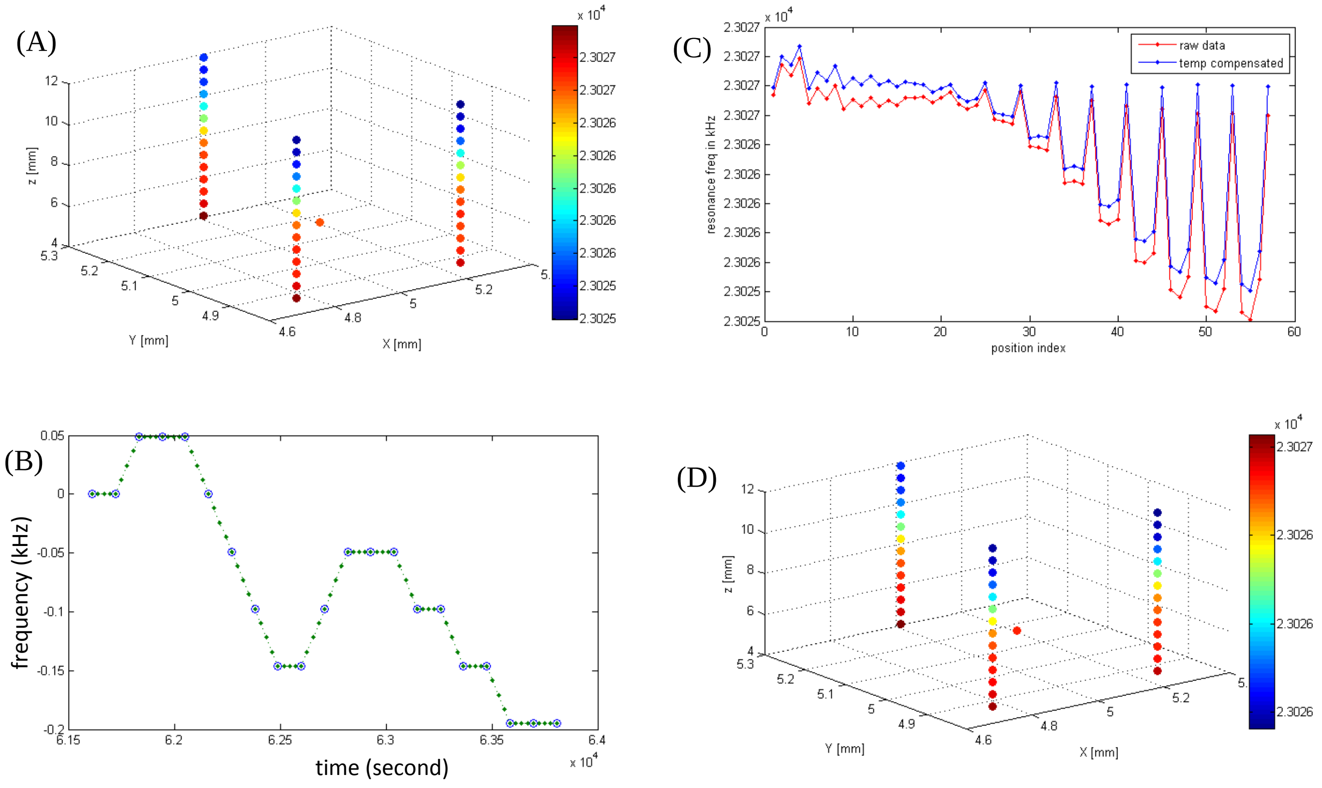


Figure 7. Field mapping for magnet M2 in the scanning pattern of Figure 6(A). (A) a map of f_0 from the Lorentzian-function fitting at all scanned points; (B) Measured (blue) and linearly interpolated (green) Larmor frequency at the reference point as a function of acquisition time; (C) Raw Larmor frequency (red) and temperature-corrected frequency (blue); (D) A map of temperature-corrected f_0 from the Lorentzian-function fitting at all scanned points.

Reference

1. "Proton precession magnetometer." U.S. Patent 3,070,745, issued December 25, 1962.
2. Specifications for the actuators can be found at https://www.newport.com/p/CONEX-TRA12CC
3. The Matlab program can be found in appendix 1.

Appendix 1

1. Scan motorized xyz stage

```
clear
close all
clc

% Parameters and constants

%! Prospa
prospaPath = 'C:\Program Files (x86)\Prospa 3.39\Prospa.exe' % full path to
executable file
sequencePath = 'C:\Users\micronmr\Desktop\1PulseBD.pex' % full path to .pex
file
outputDir = 'C:\Users\micronmr\Desktop\MFM\' % directory in which output file
is automatically saved
outputFileID = 'water'
scanCount = 2

%! XYZ stage
xport = 'COM8'
yport = 'COM9'
zport = 'COM10'
dampingTimeInSeconds = 3 % [s]

%% Import scanning position data

%! Let user select position file
%... *.csv; assume file contains (N x 3) matrix with each row = [X, Y, Z] in
millimeter
%... *.mat; assume file contains 3 vectors: [X]', [Y]', [Z]' in millimeter
filterSpec = {'*.csv'; '*.mat';};
[fileName, pathName, filterIndex] = uigetfile( filterSpec, 'Select position
data file' );
filePath = [pathName, '\', fileName];

if filterIndex == 1
    %! Read .csv file
    M = csvread( filePath );
    X = M(:, 1);
    Y = M(:, 2);
    Z = M(:, 3);
elseif filterIndex == 2
    %! Read .mat file
    load( filePath );
    assert( length( X ) == length( Y ) )
    assert( length( X ) == length( Z ) )
else
    error( 'Error: Unsupported file format.' );
end

%! Make sure the map is right
figure
plot3(X,Y,Z,'o-');grid on
xlabel( 'X [mm]' ); ylabel( 'Y [mm]' ); zlabel( 'z [mm]' )
```

```
% subplot( 1, 3, 1 ); plot( X, Y, 'o-' ); xlabel( 'X [mm]' ); ylabel( 'Y
[mm]' ); grid on
% subplot( 1, 3, 2 ); plot( Y, Z, 'o-' ); xlabel( 'Y [mm]' ); ylabel( 'Z
[mm]' ); grid on
% subplot( 1, 3, 3 ); plot( X, Z, 'o-' ); xlabel( 'X [mm]' ); ylabel( 'Z
[mm]' ); grid on

prompt = 'Do you want to proceed? Y/N [N]: ';
proceed = input( prompt, 's' );
if ~strcmpi( proceed, 'Y' )
    return
end

%% Initialize XYZ stage

%% Initialize Prospa output file
%{
prompt = 'Delete previous scan data to proceed? Y/N [N]: ';
proceed = input( prompt, 's' );
if ~strcmpi( proceed, 'Y' )
    return
end

%! Delete files
...
%}
%% Scan field
% initialize
movestage( xport, X(1) )
pause(3)
movestage( yport, Y(1))
pause(3)
movestage( zport, Z(1) )
pause(3)
%%

ptCount = length( X );
for ptNo = 1 : ptCount
    %! Move to new point
    x = X(ptNo);
    y = Y(ptNo);
    z = Z(ptNo);
    fprintf( 'move to [%d, %d, %d]', x, y, z )

    movestage(xport, x)
    pause(dampingTimeInSeconds)
    movestage(yport, y)
    pause(dampingTimeInSeconds)
    movestage(zport, z)
    pause(dampingTimeInSeconds)

    %! Wait for vibration damping


    %! Run NMR measurement
    %... the command should read like:
    %... subprocess.call('"C:\\Program Files (x86)\\Prospa\\Prospa.exe"
"C:\\Program Files (x86)\\Prospa\\Macros\\Kea-NMR\\1PulseDB.pex" "20"')
    nmrCmd = ['"', prospaPath, '" "', sequencePath, '" "',
num2str( scanCount ), '" "', num2str( ptNo ), '"'];
    system( nmrCmd );

    %! Wait for acquisition complete
    outputFilePath = [outputDir, outputFileID, '\',num2str( ptNo ), '\',
'data.csv'];
    outputExist = false;
    while ~outputExist
        outputExist = (exist( outputFilePath, 'file' ) == 2);
        pause( 1 );
    end
    disp([num2str(ptNo) ' NMR scan' ' complete'])
end %/ for ptNo

%% Finalize XYZ stage
```

2. Generate scan patterns

```
Generate scan patterns
clear
xscan = 4;
yscan = 4;
zscan = 4;
delta = 0.3 % in mm

%s = zeros(3,xscan*yscan*zscan+1);

% initial point
x0 = 5
y0 = 5
z0 = 7.5
s(:,1) = [x0 y0 z0];
%%
clear index
index = 2;
index2 = (-yscan/2);
for i = (-xscan/2):(xscan/2)
    for j= (-yscan/2):(yscan/2)
        for k = (-zscan/2):(zscan/2)
            if j == index2
                s(:,index) = [x0+delta*(i) y0+delta*(j) z0+delta*(k)]; %
update position...

            else
                s(:,index) = [x0 y0 z0]; % go back to original position for
temp calibration
                index = index + 1;
                s(:,index) = [x0+delta*(i) y0+delta*(j) z0+delta*(k)]; %
update position...

            end
            index2 = j;
            index = index + 1;
        end
    end
end
s(:,index) = [x0 y0 z0]
%
% index2 = 0;
% for i = 1:(xscan/2)
%     for j= 1:(xscan/2)
%         for k = 1:(xscan/2)
%             s(:,index2+index) = [x0-delta*(i-1) y0-delta*(j-1) z0-delta*(k-
1)];
%             index2 = index2 + 1;
%         end
%     end
% end

% export to csv
csvwrite('scantable.csv',s')
```

3. Function to move the stage, input the port name and position

```
function f = movestage(port,position)

delete(instrfind); %deletes previously declared communication ports
ads = 'D:\Newport\MotionControl\CONEX-
CC\Bin\Newport.CONEXCC.CommandInterface.dll';

A=NET.addAssembly(ads); %add Assembly
%
% Choose the desired stage to move:

obj = port; % x axis
%obj = 'COM4'; % y axis
%obj = 'COM5'; % z axis


comport = serial(obj); %define the serial object. For z axis

%
CC = CommandInterfaceConexCC.ConexCC(); %set CC to the library.class
R = CC.OpenInstrument(obj); % open the cp

[a,b,c] = CC.SA_Get(1); % get the controller address
[a,b,c] = CC.TP(1); % get current position in b
[a,b,c] = CC.AC_Get(1); % get current acceleration in b
[a,b,c] = CC.VA_Get(1); % get current velocity in b

CC.VA_Set(1,0.4); % update velocity
[a,b,c] = CC.VA_Get(1);
CC.AC_Set(1,1.2); % update acceleration
[a,b,c] = CC.AC_Get(1);
```

```
CC.MM_Set(1,1); % Set the system to READY state
%% Momve the motor to absolute position

CC.PA_Set(1,position); % update absolute position
%% Close the port
CC.CloseInstrument();
end
```

4. Data analysis

```
% load data into matlab
clear
clc
%%
nrpts = 151;
rf = 23.0268*10^3; % in kHz
for i = 1:nrpts
    spc(i,:,:) = csvread(['C:\Users\YTang12\Desktop\Reference\Data\Magnetic
field mapping\Water\' num2str(i) '\spectrum.csv'])';
    FileInfo = dir(['C:\Users\YTang12\Desktop\Reference\Data\Magnetic field
mapping\Water\' num2str(i) '\spectrum.csv']);
    TimeStamp = FileInfo.date;
    a = datevec(TimeStamp);
    timebase(i) = 3600*a(4) + 60*a(5) + a(6);
end

% load the position scantable
scantable = csvread(['C:\Users\YTang12\Desktop\Reference\Data\Magnetic field
mapping\scantable.csv']);
% load temp reference index
for i = 1:26
    TimeRefInd(i) =  (i-1)*6 + 1
end

%% Find the resonance frequency for each acquisition
% ind is a rough determination of resonance frequency
freqbase = spc(1,1:2:end) % in kHz
for i = 1:nrpts
    %subplot(13,13,i)
    [value(i),ind(i)] = max(spc(i,2:2:end))
end
plot(value)
figure(29)
plot(freqbase(ind(1:6:end)),'.-') %freq at the time ref points

%plot(spc(149,2:2:end),'.')
ind(149) = 1025;
plot(freqbase(ind(1:end)),'.-') %freq for all acquisitions
%% Plot all FFT spectrum and the resonance frequency in kHz
figure
for i = 1:nrpts
   % subplot(1,2,1)
    title(num2str(i))
    subplot(13,13,i)
    plot(freqbase,spc(i,2:2:end),'.')
   % ylim([0 50])
   % subplot(1,2,2)
   % plot(freqbase(ind(i)),'.-')
   % pause
end
%% calculate linewidth
% fit the data with a Lorentzian function
f=@(a,x)(a(1) /pi*a(2)./((x-a(3)).^2+(0.5*a(2))^2));

a=[20,1,1];
window = 40;

for i = 1:nrpts
    subplot(13,13,i)
    [ahat(i,:),r,J,cov,mse] = nlinfit(freqbase((ind(i)-
window/2):(ind(i)+window/2)),spc(i,(ind(i)-
window/2)*2:2:(ind(i)+window/2)*2),f,a);
    ylim([0 50])
    title(num2str(i))
    plot(freqbase((ind(i)-window/2):(ind(i)+window/2)),spc(i,(ind(i)-
window/2)*2:2:(ind(i)+window/2)*2),'b.')
    hold on
    plot(freqbase((ind(i)-
window/2):(ind(i)+window/2)),f(ahat(i,:),freqbase((ind(i)-
window/2):(ind(i)+window/2))),'r');
    hold off
%    pause
end
%% ahat(:,3) is the resonance frequency from the fitting

% now convet ahat(:,3) to the nearst index in the frequency base
for i = 1:nrpts
    clear temp
    for j = 1:2048
        temp(j) = abs(freqbase(j) - ahat(i,3));
        [value2(i),ind2(i)] = min(temp);
    end
end

plot(1:151,ahat(:,3),'.-r')
hold all
plot(1:151,freqbase(ind(:)),'.-b')
plot(1:151,freqbase(ind2(:)),'.-c')


%% Do temperature normalization
% We assume that the field homogeneity is independent of temperature. Only
% a simple golbal scaling factor is considered
TempCorr =
interp1(timebase(TimeRefInd),freqbase(ind2(TimeRefInd)),timebase) %TempCorr
is the baseline for normalization for all measurements
plot(timebase(TimeRefInd),freqbase(ind2(TimeRefInd)),'o',timebase,TempCorr,':
.')
absfreq = freqbase + rf;
plot(absfreq)
```

```
% calculate normalized B field in the scanned cell.
% B = B0*(1 + delta), where delta is the temperature coefficient.
% Then delta = B/B0 - 1
delta = (rf-(TempCorr - TempCorr(1)))/rf-1
nabsfreq = absfreq(ind2(1:end)).*(1+delta);
subplot(2,2,2)
plot(nabsfreq,'r.-')
hold all
plot(absfreq(ind2(1:end)),'b.-')
ylabel('resonance freq in kHz')
xlabel('position index')

% plot scanned volume in space
X = scantable(:,1);
Y = scantable(:,2);
Z = scantable(:,3);

%plot3(X, Y, Z,'o');grid on
subplot(2,2,4)
scatter3( X(1:151), Y(1:151), Z(1:151), [], nabsfreq(1:151), 'filled' )
%isosurface(X,Y,Z,absfreq)
xlabel( 'X [mm]' ); ylabel( 'Y [mm]' ); zlabel( 'z [mm]' );

subplot(2,2,3)
plot(timebase(TimeRefInd),freqbase(ind2(TimeRefInd)),'o',timebase,TempCorr,':
.')

subplot(2,2,1)
scatter3( X(1:151), Y(1:151), Z(1:151), [], absfreq(ind2(1:end)), 'filled' )
%isosurface(X,Y,Z,absfreq)
xlabel( 'X [mm]' ); ylabel( 'Y [mm]' ); zlabel( 'z [mm]' );

%% ahat(2) is the linewidth of the Lorentzian peak
plot(abs(ahat(:,2)))
for i = 1:nrpts
    inhomo(i) = abs(ahat(i,2))/absfreq(ind(i))
end

scatter3( X(1:149), Y(1:149), Z(1:149), [], inhomo(1:149), 'filled' )
%isosurface(X,Y,Z,absfreq)
xlabel( 'X [mm]' ); ylabel( 'Y [mm]' ); zlabel( 'z [mm]' );
```